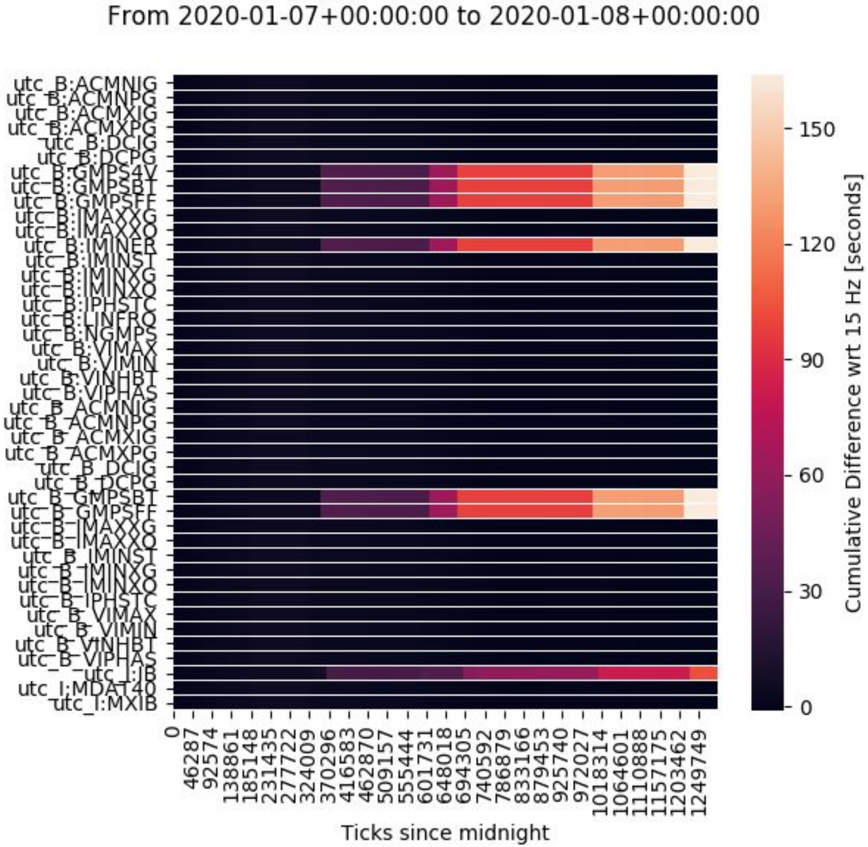
Time Stamps in Data Files

Jason St. John
2019.01.13
Accelerator AI Group

Compare to naive 15 Hz reference. Example: Jan 7



For all parameters in the data, compare time stamps to a naive, 64-bit-precise 15 Hz reference.

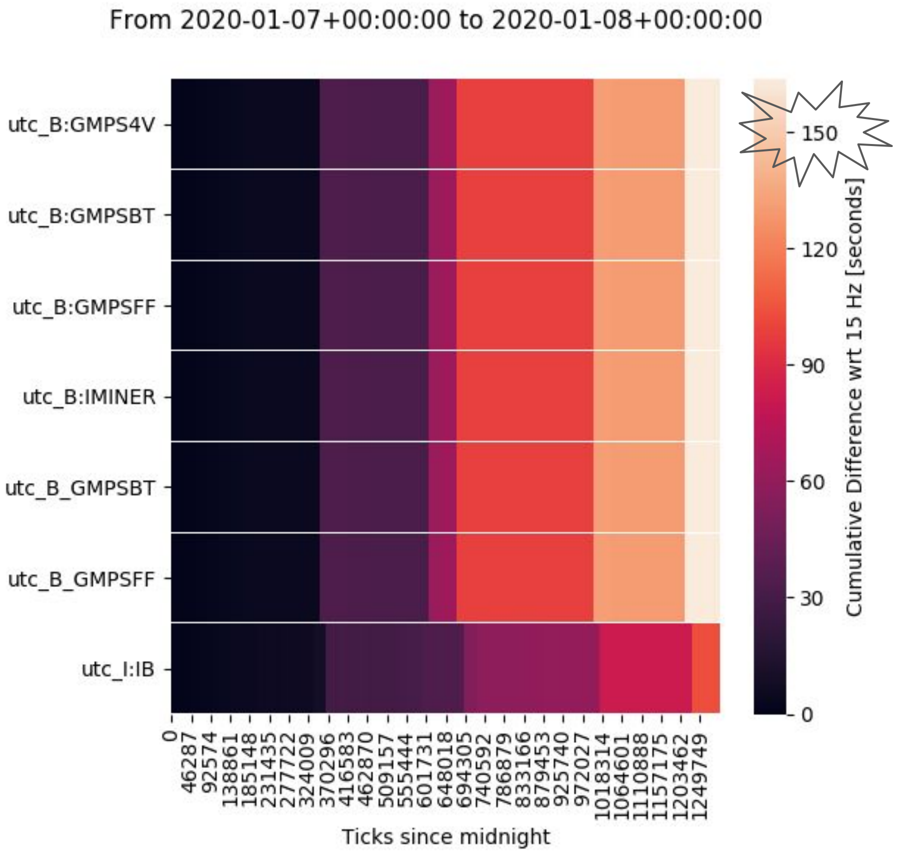
Certain **outlier** parameters' variation dominates, setting the color scale.

<u>>150 s in 24 hr</u>

Hard to see any structure to the **inliers**.

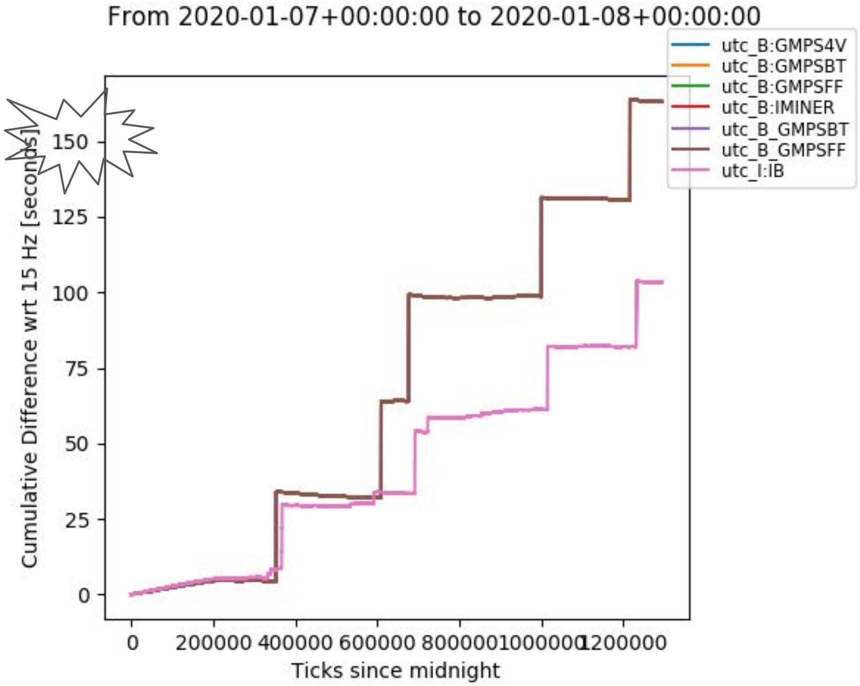
Following is an initial investigation into this behavior in our data.

Compare to naive 15 Hz reference. Example: Jan 7

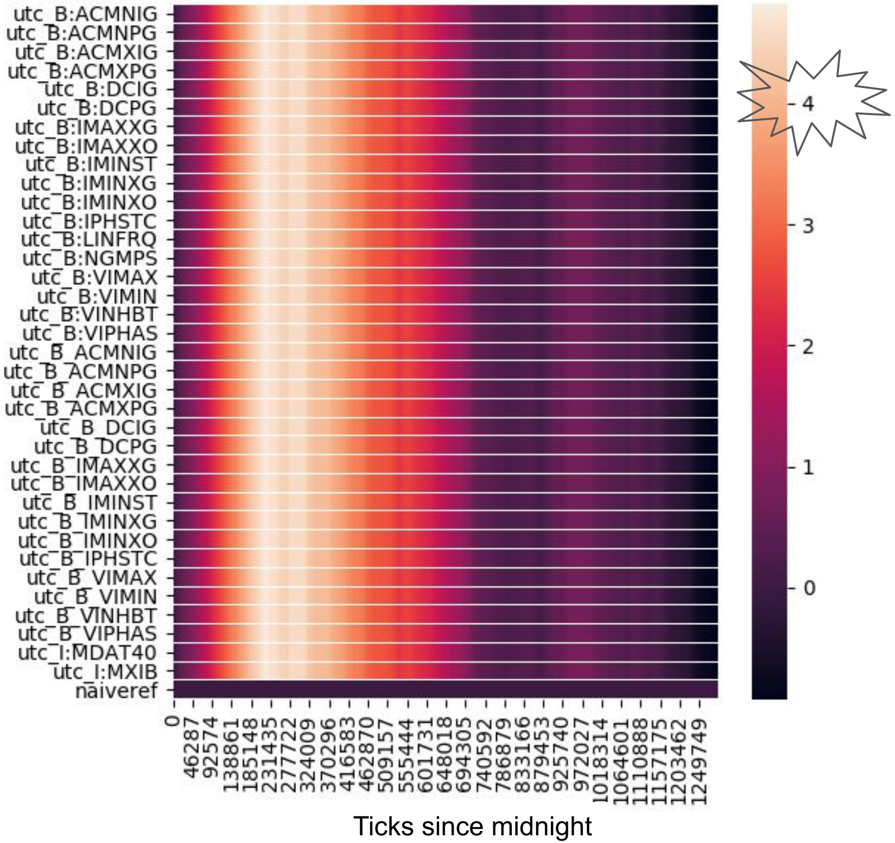


Outliers only. Collective jumps (I:IB solo)

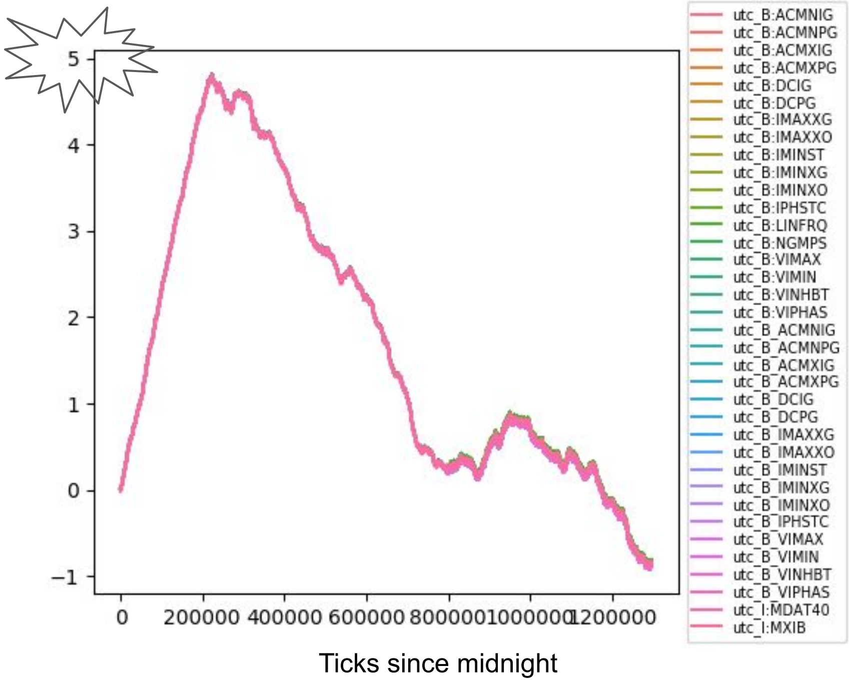
Also as line plot:



Compare to naive 15 Hz reference. Example: Jan 7



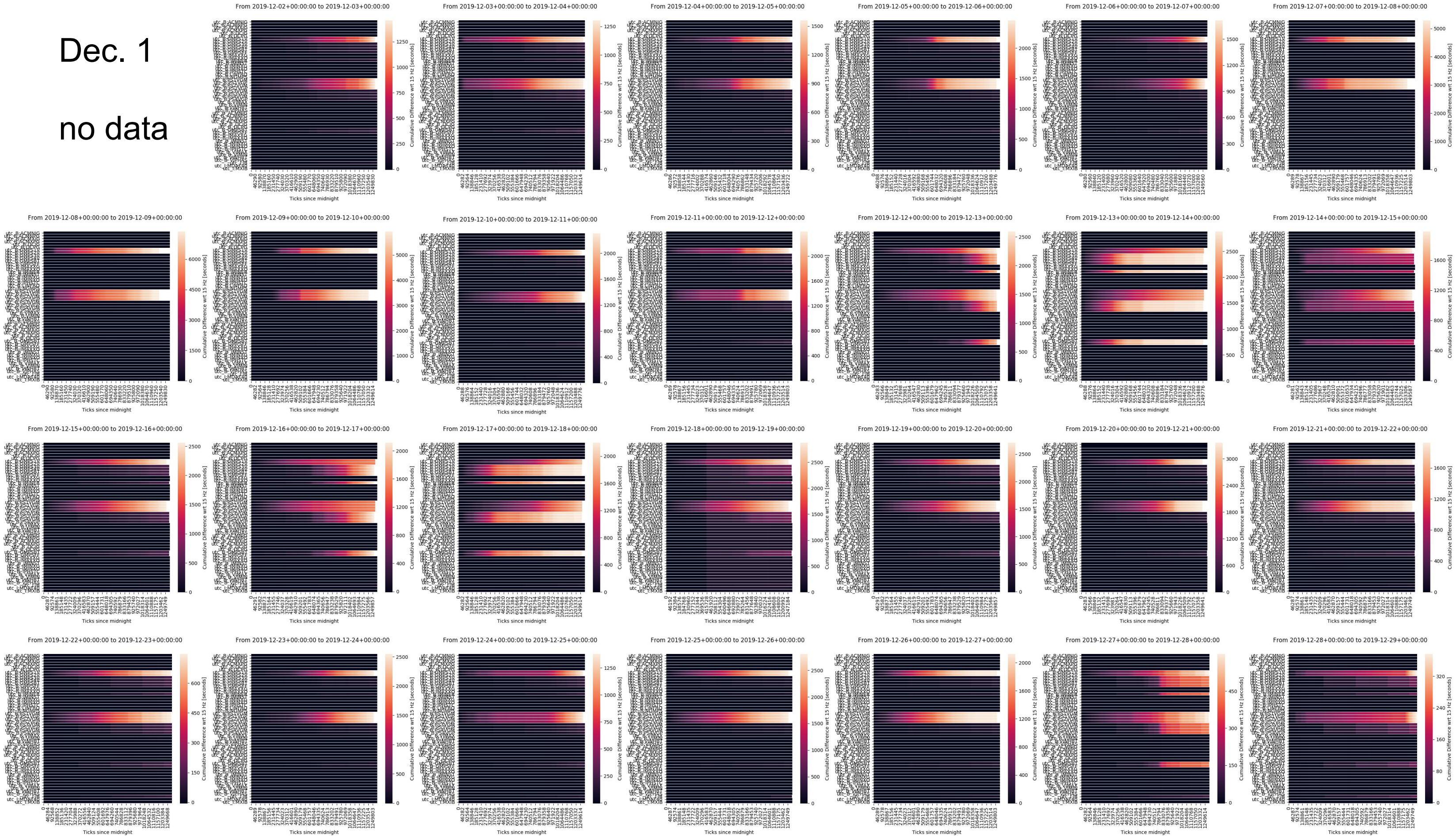
Inliers only. Collective drifting.
Also as line plot:



OK was that a typical day? Here’s 12-1 to 12-30

Dec. 1

no data



From 2019-12-02+00:00:00 to 2019-12-03+00:00:00

From 2019-12-03+00:00:00 to 2019-12-04+00:00:00

From 2019-12-04+00:00:00 to 2019-12-05+00:00:00

From 2019-12-05+00:00:00 to 2019-12-06+00:00:00

From 2019-12-06+00:00:00 to 2019-12-07+00:00:00

From 2019-12-07+00:00:00 to 2019-12-08+00:00:00

From 2019-12-08+00:00:00 to 2019-12-09+00:00:00

From 2019-12-09+00:00:00 to 2019-12-10+00:00:00

From 2019-12-10+00:00:00 to 2019-12-11+00:00:00

From 2019-12-11+00:00:00 to 2019-12-12+00:00:00

From 2019-12-12+00:00:00 to 2019-12-13+00:00:00

From 2019-12-13+00:00:00 to 2019-12-14+00:00:00

From 2019-12-14+00:00:00 to 2019-12-15+00:00:00

From 2019-12-15+00:00:00 to 2019-12-16+00:00:00

From 2019-12-16+00:00:00 to 2019-12-17+00:00:00

From 2019-12-17+00:00:00 to 2019-12-18+00:00:00

From 2019-12-18+00:00:00 to 2019-12-19+00:00:00

From 2019-12-19+00:00:00 to 2019-12-20+00:00:00

From 2019-12-20+00:00:00 to 2019-12-21+00:00:00

From 2019-12-21+00:00:00 to 2019-12-22+00:00:00

From 2019-12-22+00:00:00 to 2019-12-23+00:00:00

From 2019-12-23+00:00:00 to 2019-12-24+00:00:00

From 2019-12-24+00:00:00 to 2019-12-25+00:00:00

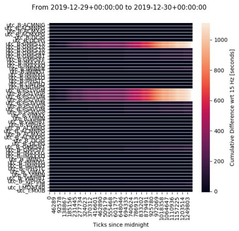
From 2019-12-25+00:00:00 to 2019-12-26+00:00:00

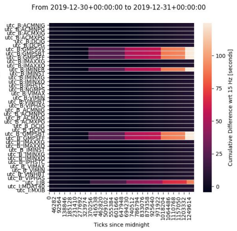
From 2019-12-26+00:00:00 to 2019-12-27+00:00:00

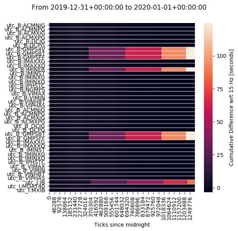
From 2019-12-27+00:00:00 to 2019-12-28+00:00:00

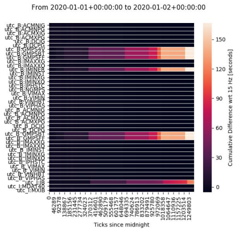
From 2019-12-28+00:00:00 to 2019-12-29+00:00:00

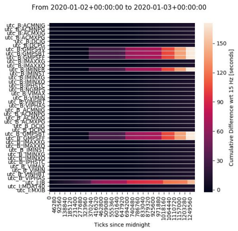
OK was that a typical day? Here’s 12-1 to 12-12

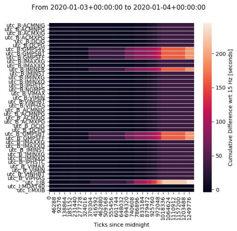

From 2019-12-29+00:00:00 to 2019-12-30+00:00:00

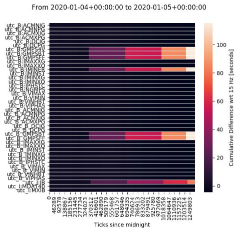

From 2019-12-30+00:00:00 to 2019-12-31+00:00:00

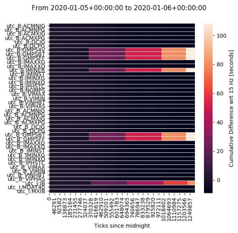

From 2019-12-31+00:00:00 to 2020-01-01+00:00:00

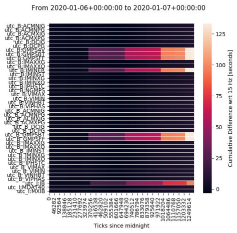

From 2020-01-01+00:00:00 to 2020-01-02+00:00:00

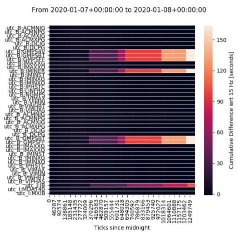

From 2020-01-02+00:00:00 to 2020-01-03+00:00:00

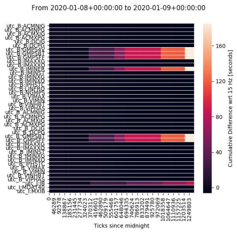

From 2020-01-03+00:00:00 to 2020-01-04+00:00:00

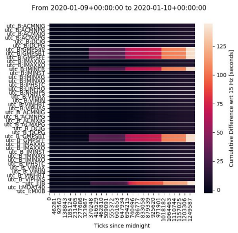

From 2020-01-04+00:00:00 to 2020-01-05+00:00:00

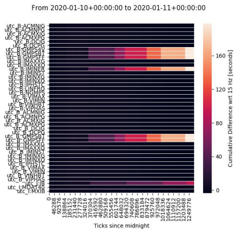

From 2020-01-05+00:00:00 to 2020-01-06+00:00:00

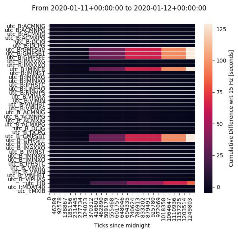

From 2020-01-06+00:00:00 to 2020-01-07+00:00:00

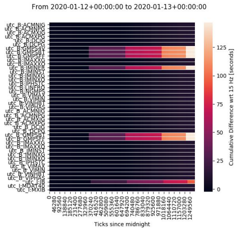

From 2020-01-07+00:00:00 to 2020-01-08+00:00:00


From 2020-01-08+00:00:00 to 2020-01-09+00:00:00


From 2020-01-09+00:00:00 to 2020-01-10+00:00:00


From 2020-01-10+00:00:00 to 2020-01-11+00:00:00

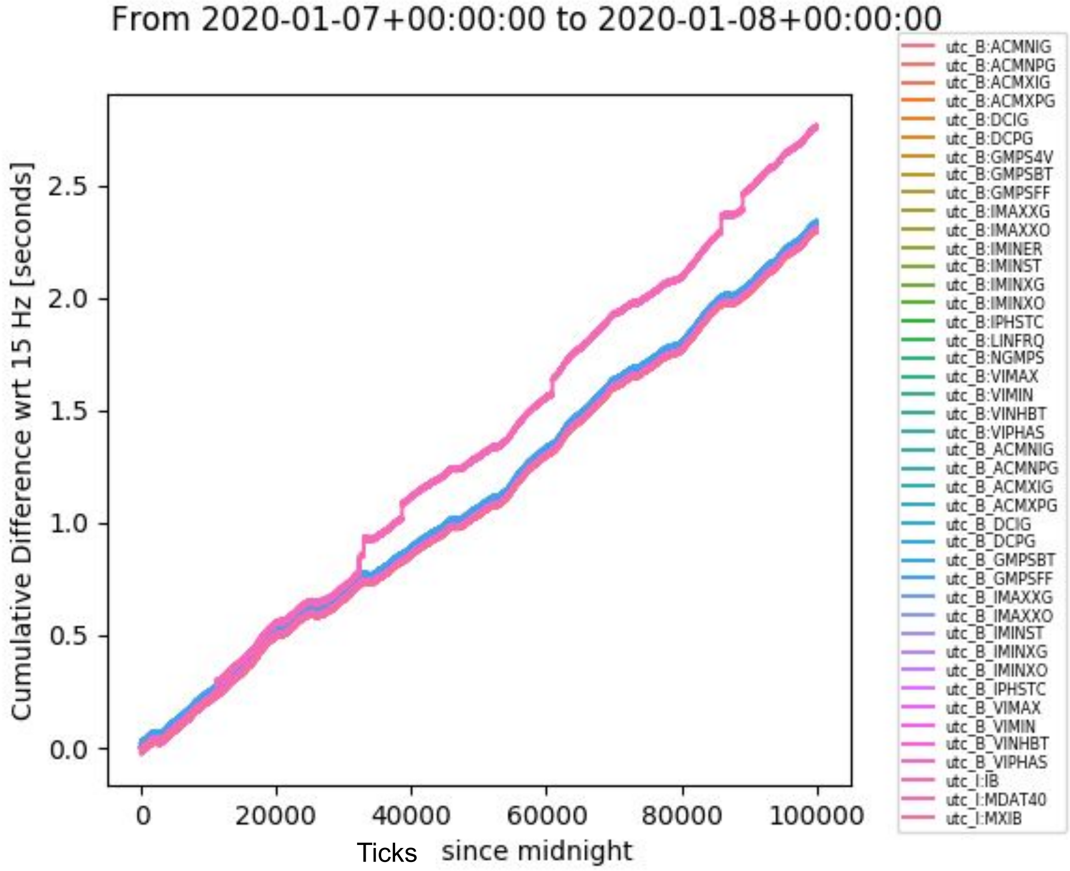

From 2020-01-11+00:00:00 to 2020-01-12+00:00:00


From 2020-01-12+00:00:00 to 2020-01-13+00:00:00

Some parameters’ timestamps show jumps only some days, others jumps are the norm (but not the rule).
→ Possible causes?

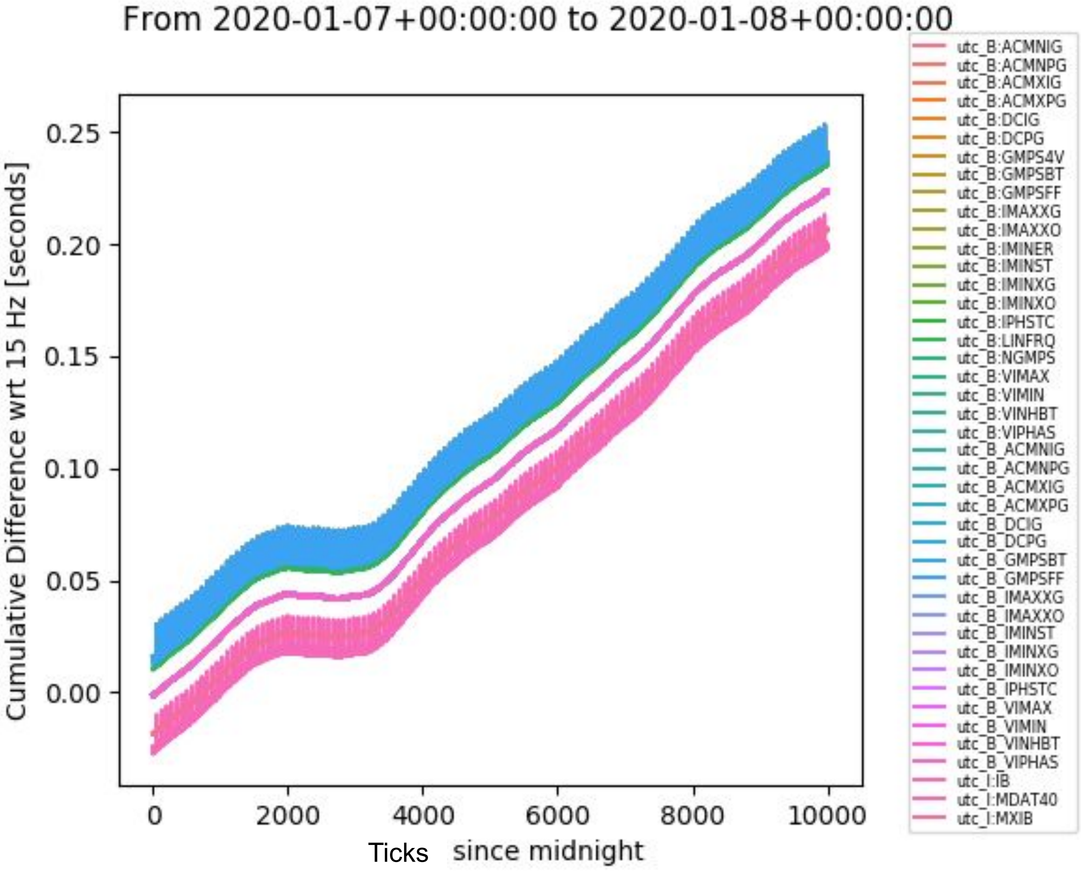
How would we (/should we) best keep tabs on this, in an automated monitoring system?

Fine time structure, too: (Example: Jan 7)



100,000 seconds

Fine time structure, too: (Example: Jan 7)

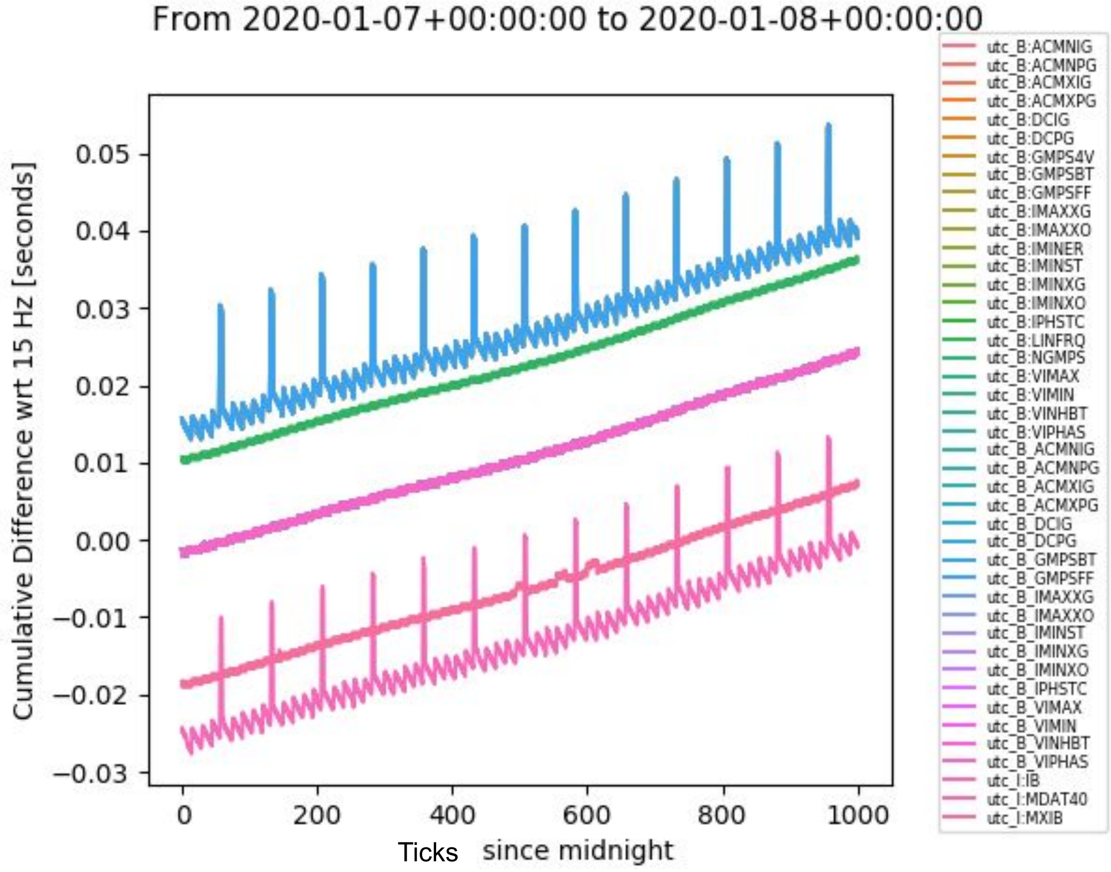


10,000 seconds

Collective drift has impact on all parameters, even those subject to apparent jumps.

So how are the jumping channels special?

Fine time structure, too: (Example: Jan 7)



1,000 seconds

Periodic spikes (jump, then reversal) <u>~0.015 s</u> lasting several 15 Hz ticks

Plus quicker sawtooth <u>~0.002 s</u>

Implications for operating (and training!) LSTM nets

- Jumps & drifts: Entirely an artifact of how Logger & whole system work together?
    - *Or* would an online data customer really see any of this variation in parameter value update times?
    - Want to get it right:
      Net training environment should mimic operation environment closely.
- At the very least, we could feed the time (epoch µs for instance) as part of the input vector?
- Does the value of B:LINFRQ predict this? Especially the slow drifts?
    - Improve realism using timestamp of $0C?
- Implications for using Param data and Event timing:
    - Event data also timestamped (plus ms since $00 event)
    - Can we take event timestamps at face value when training nets with events & parameter as inputs?